

SYSTEM AND METHOD FOR MANAGING SYSTEM
CONFIGURATION ACROSS A NETWORK

Related Application

This application is a continuation-in-part claiming the benefit of U.S. Patent Application Serial Number 08/814,771, filed on March 7, 1997 and to be issued as U.S. Patent No. 6,074,434, which is itself a continuation-in-part of U.S. Patent Application Serial Number 08/659,841, filed on June 7, 1996, and issued as U.S. Patent No. 5,752,042.

Field of the Invention

The present invention relates generally to computer systems and, more specifically, to system configuration management for a plurality of computers connected via a network.

Background of the Invention

The use of the client-server distributed computing model in local area networks and enterprise-wide systems has become ubiquitous in business, academic, and personal computing environments. With increasing frequency, fixed function display terminals have been replaced by desktop and laptop computers capable of executing software programs independently and asynchronously from the centralized mainframe or enterprise server processor.

As the number of desktop and laptop computers in a local (or enterprise-wide) network increases, the complexity of maintaining the software configuration in each computer increases. Software developers frequently distribute updated versions or releases of their software products, referred to as "updates" in this document, to correct software errors or to enhance functionality, compatibility, and portability. Of course, software developers also distribute new software products designed to improve performance, typically with an increase in the complexity of installing and configuring such new software products.

There are various known techniques for a server to provide a client with a code update or a user with new data. In one technique, clients register with the server for updates to a particular program and users register with the server for updates or new data. Whenever the server obtains a new update for that program or new data of the specified type, the server automatically sends the code update or new data of the specified type to every client/user on the list. Unfortunately, some users may not wish to obtain a particular update or particular new data of the specified type because of cost, download time, or other reason. Alternately, a user can periodically request all code updates or new data of specified types and the server will respond accordingly. Unfortunately again, the user may not need every code update or data of the specified type but, nevertheless, endures the cost and download time. In general, because of the large number and complexity of software applications on a user's system, and the large number and complexity of the corresponding updates for that system, it is very difficult for users to know what updates are appropriate for them. Likewise, there is so much new data generated every day that it is difficult to know what new data is appropriate for particular users.

Thus, network operators have been faced with the challenge of balancing their desire to provide their clients with the most current software updates and products against their requirement of minimizing disruptions to the clients. Software configuration management tools have been developed to assist network operators in this endeavor.

U.S. Patent No. 5,752,042 issued to Cole et al. discloses a server computer and a method performed by a server computer for selecting code updates to download to a client computer. The server computer identifies code updates which are consistent with basic system characteristics of the client computer. Then, the server computer sends to the client a "recognizer" program which executes in the client computer to determine if the client computer has a version of code that is not current. The server may then generate a list of code updates based upon the result of executing the recognizer program. A user at the client computer end can select from the list. Based on the selections made by the user, the server computer sends addresses of the selected code updates to the client computer. The client computer may then download the selected updates from a server computer.

U.S. Patent No. 5,581,764 issued to Fitzgerald et al. discloses an automated method of managing changes in a distributed computing environment. The resource needs of individual distributed computers are determined based upon a Should Have (SH) list schema. The SH list schema is used to produce SH lists for individual
5 distributed computers. Individual Already Have (AH) lists may be stored or produced at configuration time for individual distributed computers. A differencing process is used to locate differences between SH lists and AH lists. The differences between SH lists and AH lists for the individual distributed computers are used to determine what items must be updated or modified in order to update individual desktops.

10 In prior art configuration management systems, information describing the software configuration attributes of each client processor was provided to a central server processor. The central server processor would determine which updates and new products to provide to each client, based on the software configuration of each client.

15 There remains a need, therefore, for a more flexible and robust software and hardware configuration management system.

Summary of the Invention

To meet this and other needs, and in view of its purposes, the present invention provides a system and method for determining whether to provide a software program update to a client processor. A configuration receiving mechanism receives and
20 stores a set of hardware, software, or both hardware and software (or "system") configuration attributes from the client processor. The system configuration attributes indicate whether a copy of a client software program may be stored in the client processor. A configuration transmitting mechanism transmits the set of system configuration attributes to a server processor. An update-receiving mechanism receives
25 an update recognizer program and a software program update from the server processor.

The update recognizer program and the software program update are associated with the client software program. An update recognizer transmitting mechanism transmits the update recognizer program to the client processor. The client processor executes the update recognizer program to issue a notification indicating

whether the software program update is applicable to the copy of the client software program which may be stored in the client processor.

It is to be understood that both the foregoing general description and the following detailed description are exemplary, but are not restrictive, of the invention.

5

Brief Description of the Drawing

The invention is best understood from the following detailed description when read in connection with the accompanying drawing. It is emphasized that, according to common practice, the various features of the drawing are not to scale. On the contrary, the dimensions of the various features are arbitrarily expanded or reduced
10 for clarity. Included in the drawing are the following figures:

Fig. 1 illustrates a computer network including clients, a selection server, and a content server;

Fig. 2 is a block diagram illustrating more detail of a selection server, content server, and a client in the network of Fig. 1;

15

Figs. 3a and 3b form a flow chart illustrating operation of the client and server of Fig. 2;

Fig. 4 is a flow chart illustrating a program sent from the server to the client of Fig. 2 to determine needs of the client for a particular code update;

20

Fig. 5 is another flow chart illustrating a program sent from the server to the client of Fig. 2 to determine needs of the client for another particular code update;

Fig. 6 illustrates another embodiment of a computer network;

Fig. 7 is a block diagram illustrating more detail of a selection server, content server, and a client in the network of Fig. 6;

DETAILED DESCRIPTION

Figs. 8a and 8b form a flow chart illustrating operation of the client and server of Fig. 7;

Fig. 9 is a flow chart illustrating a program sent from the server to the client of Fig. 7 to determine needs of the client for a particular set of new data;

5 Fig. 10 is another flow chart illustrating a program sent from the server to the client of Fig. 7 to determine needs of the client for another particular set of new data;

Fig. 11 is a block diagram of an exemplary system in accordance with the present invention; and

10 Fig. 12 is a block diagram of the exemplary administration server shown in FIG. 11.

Detailed Description of the Invention

Referring now to the figures in detail, in which like reference numbers indicate like elements throughout, Fig. 1 illustrates a computer network generally designated 10 including clients, a selection server, and a content server. Network 10 comprises a selection server 12 with a data base 13, a content server 17, clients 14-16, and an Internet 20 to interconnect the servers and clients. By way of example, the selection server 12 comprises a computer workstation with associated programming and a modem, the content server 17 comprises a computer workstation with associated programming and a modem, and each of the clients 14-16 comprises a personal computer with associated programming and a modem. The Internet 20 comprises a communication medium such as telephone wires between the clients 14-16, the selection server 12, and the content server 17 and programming to manage the communications.

25 Communications between the clients 14-16 and the selection server 12 use hypertext transport protocol (HTTP) and communications between the clients 14-16 and the content server 17 use file transport protocol (FTP). Although not shown, there are many other servers for the Internet. Selection server 12 is dedicated to automating selection of code updates, the content server 17 includes the actual code updates (stored

on disk 21), and data base 13 includes meta data for the code updates and FTP addressing information to locate the code updates in content server 17. The meta data provides descriptive information about the respective code updates such as basic input output system (BIOS) level, pre-load level, motherboard ID, and the like. The FTP addressing 5 information indicates the address or identification of the content server which stores the respective code update (content server 17 in the illustrated example), a user ID and password to log-on to the content server, a name of the code update, a list of files which comprise the code update and the size, checksum, and directory of each file.

According to one process for storing the updates in the content server 17
10 and the meta data in the selection server 12, a person writes the code update and meta data and loads them in the content server 17, either via the Internet or by manual loading. The code updates remain stored in the content server, but the meta data is written to the selection server, either via the Internet or by manual loading.

As illustrated in Fig. 2, the selection server 12 is an HTTP server with an
15 associated "CGI" program manager 31 and includes an update selection program 30. The client 14 includes an update manager program 32 (including a graphical user interface or "GUT"), a scout routine 33, a service application routine 34, and a download routine 39. The functions of the server and client programming and a typical client-server session for selecting and downloading updates are illustrated in Figs. 3a and 3b. A user at client
20 computer 14 selects an icon to invoke update manager 32. In response, the update manager contacts the general manager 31 in server 12 to begin a session and supplies the current level of update manager 32, scout 33, service application 34, and download routine 39 within client 14 (step 104). In response, the general manager 31 determines if the client's update manager 32, scout 33, service application 34, and download routine 39
25 are the latest version (step 106). This determination is performed by comparing the client level information supplied by the client 14 to the data in selection server 12 for the latest version of the client's update manager 32, scout 33, service application 34, and download routine 39 stored in the content server 17.

If the client's update manager 32, scout 33, service application 34, and
30 download routine 39 are not the latest version, the server 12 sends the FTP addressing information to the client 14 to allow the client to retrieve the latest versions from a

DRAFT - DECODED

content server 17. Along with the FTP addressing for the client update manager, scout, service application, and download routine, the server 12 sends FTP addressing information to the client 14 for a basic system information recognizer program 41 stored in content server 17 (step 107). Next, the client 14 downloads and installs the latest 5 version of the client update manager 32, scout 33, service application 34, and download routine 39 from the content server 17 if the FTP addressing was provided (step 108).

The furnishing of the FTP addressing information for the basic system information recognizer program begins the selection process. The client 14 next 10 downloads the basic system information recognizer program 41 from the content server 17 and executes it (step 109). The basic system information recognizer program 41 then obtains basic system information, using scout APIs, about client 14 (step 109). The basic system information comprises system model, pre-load software level, BIOS level, and information that is not likely to change often such as type of operating system. For example, the basic system information of client 14 is BIOS level 123, BIOS date 1/1/96, 15 and mother board ID XYZ. The update manager 32 then sends the basic system information, obtained by the basic system information recognizer program 41, to the selection server 12 (step 110).

This initiates the selection update program 30 (step 112). The selection update program 30 within server 12 then determines which code updates are consistent 20 with the basic system information of the client and which code updates are inconsistent, thereby eliminating the vast majority of the code updates stored in content server 17 (and other content servers, if any) from further consideration (step 114). In the illustrated embodiment, this determination is made by correlating the meta data in data base 13 to the basic system information obtained from client 14. Matches between the meta data 25 and the basic system information indicate that the corresponding code updates are potentially appropriate for client 14. (In practice, the meta data can be arranged in a relational data base to facilitate the correlation.) The two columns of the following table list code updates available in content server 17 and the meta data for these code updates. The two columns of this table are used for the initial correlation of the basic system 30 information obtained from client 14 to the meta data (column 2) stored in selection server 12.

	<u>Code Updates</u>	<u>Basic System Information/Meta Data</u>
	Device Driver	BIOS Level 123
	ABCDE.DRV	BIOS date 1/1/96 Mother Board ID XYZ
5	Netcomber 2.0	BIOS Level 123 BIOS date 1/1/96 Mother Board ID XYZ
10	Device Driver	BIOS Level 456
	FGHIJ.DRV	BIOS date 2/1/96 Mother Board ID XYZ
15	In this example, the BIOS level 123, BIOS date 1/1/96, and mother board ID XYZ basic system information obtained from client 14 is consistent with a device driver file named ABCDE.DRV. and Netcomber (trademark of International Business Machines Corporation) version 2.0. Device driver file named FGHIJ.DRV requires BIOS level 456, however, so is inconsistent with the basic system information obtained from client 14 and is eliminated from further consideration.	
20	For each code update, such as device driver file named ABCDE.DRV. and Netcomber program version 2.0, which is consistent with the basic system information of client 14, the selection update program 30 sends to the client 14 the FTP addressing information of a corresponding recognizer program, such as recognizer programs 40 and 42, respectively (step 116). The FTP addressing information is not provided (and no	
25	recognizer program is fetched) to client 14 for the device driver file named FGHIJ.DRV, however, because file FGHIJ.DRV is not consistent with the basic system information of client 14. From the FTP addressing information, the client 14 downloads the actual recognizer programs from the content server 17 (step 118).	

Each recognizer program is executable and small, typically 2,000 - 3,000 bytes. Consequently, each recognizer program can be downloaded from content server 17 to the client 14 in seconds (whereas the corresponding code update is much larger and may take hours to download).

It should be noted that, based on the basic system information alone, the update selection program 30 within server 12 is not sure which of the code updates that are consistent with the basic system information are actually needed by the client. For example, the client 14 may already have the latest version.

5 Next, the client 14 executes each recognizer program 40 and 42 with assistance from the scout 33 (step 118). The scout 33 assists the recognizer programs by providing subroutines that can be called by the recognizer programs. For files required by the recognizer programs, these callable subroutines can locate the files, determine the version of the files, determine the length of the file, identify a directory which lists the
10 files, obtain time stamps for the files, obtain data from inside the files, and obtain data from a program registry. This further reduces the requisite length of each recognizer program and therefore shortens the download time. When executed, each recognizer program further investigates the hardware, software, and other components of the client 14 for information to determine whether the corresponding code update is appropriate for
15 the client 14. For example, this investigation reveals whether the client 14 already has these code updates. A recognizer program could also determine if the corresponding code update is consistent with aspects of the client other than the basic system information, such as other programs within the client.

Recognizer program 40 determines from the routine of scout 33 in client
20 14 if the size of file ABCDE.DRV, stored by the client 14 is the same as the size of file ABCDE.DRV, stored in the data base 13. If so, then client 14 has the latest update. If not, then client 14 likely has an old version which could be updated.

Fig. 4 illustrates recognizer program 40 in more detail. In step 200, recognizer program 40 calls scout 33 to scan the hard disk of client 14 for file
25 ABCDE.DRV to determine if file ABCDE.DRV exists in client 14. If not, the recognizer program provides an indication that file ABCDE.DRV is not appropriate for client 14 (step 202). If file ABCDE.DRV exists in client 14, however, then recognizer program 40 calls scout 33 to retrieve the size of file ABCDE.DRV in client 14 (step 204), and then compares the file size to the size of the latest version stored in the content server
30 (decision 206). (The recognizer program includes an indication of the size of the corresponding update in the content server.) If the two sizes are the same, then there is

00000000000000000000000000000000

no need to download file ABCDE.DRV from the content server 17; client 14 already has the latest version. If the two sizes are not the same, then recognizer program 40 makes a record that the file ABCDE.DRV stored in the content server 17 is appropriate for client 14 (step 208).

5 Recognizer program 42 compares a version number of the Netcomber program in the client to the version number of the Netcomber program of the code update, and if the version number of the Netcomber program installed in client 14 is less, then the client likely has an old version. Fig. 5 illustrates recognizer program 42 in more detail. In step 300, recognizer program 42 calls scout 33 to read a data base in client 14
10 to determine if the Netcomber program exists in client 14. If it does, recognizer program 42 calls scout 33 to obtain the version number of the Netcomber program from the data base (step 302). If the version number stored in the data base is equal to or greater than the version number in the recognizer program, then client 14 does not need the version stored in the content server 17 (decision 304 and step 306). If the version number stored
15 in the data base is less than the version number in the recognizer program, however, then the recognizer program records that client 14 needs the copy of the Netcomber program stored in the content server 17 (decision 304 and step 308).

20 The results of the recognizer programs include yes/no answers to the question of whether each respective code update is appropriate for the client, i.e. not currently resident in client 14 and consistent with the client basic system information as determined previously (in step 114). The recognizer programs can also assign a "critical" level to each code update based on the need of the client for the update. For example, if another program in the client needs this code update to function, then this code update would be assigned a high level of criticality. After the recognizer programs are executed
25 in client 14, client 14 sends to selection server 12 a list of the code updates which are appropriate for the client (step 120).

30 Based on the information gathered by the recognizer programs, the update selection program determines the level of criticality of the respective code updates and builds a selection form for display at the client (step 122). The selection form comprises a list of the code updates that are consistent with the basic system information and

appropriate for the client 14, and a display of the following three selection choices (step 124):

1. minimum number of code updates - code updates which are necessary for the client to ensure compatibility between programs within the client and fix a significant "bug" in a program within the client, and those other updates that the author of the code update deemed critical such as replacement of unsupported programs;
- 5 2. maximum number of code updates - all code updates which are consistent with the basic system information and appropriate for the client; and
- 10 3. client selection of specific code updates - user selects particular code updates from the list of all code updates which are consistent with the basic system information and appropriate for the client.

15 Next, the client makes a selection, either selection of the first or second categories or an itemized list of code updates pursuant to the third category (step 130), and sends the selection to the server (step 131). In response, the server 12 sends to the client 14 the FTP addressing information for the selected code updates (step 132). Using this FTP addressing information, the download routine 39 of the client downloads the code updates from the content server 17 (step 133). In a preferred embodiment, the FTP 20 addressing information for all code updates represented in the selection form are passed with the selection form to the client 14 (in step 124).

25 As the code updates are being downloaded, the download routine also records status information (such as the FTP addresses) of the code updates in case of interruption to the communication line. If there is such an interruption, the downloading can continue later where it left off. Because the interruption could be lengthy and the system characteristics could have been altered, immediately before the selected code updates are actually installed the update manager 32 invokes the corresponding recognizer programs again to ensure that the code updates are still required; it is possible that changes could have occurred to the client since the recognizer programs were

previously executed. If the code updates are still appropriate, the service application routine of client 14 installs the code updates in the client 14 as follows (step 134). For every code update file whose stale version is not currently in use by client 14, the service application routine replaces the stale file with the updated file. The update manager also

5 lists each code update file whose stale version is currently in use. Then, the client is requested to re-boot, and the operating system installs the listed code updates during the re-boot. For all code updates selected by the user that represent one or more bytes of a file, the service application routine replaces just the one or more bytes of the file. For such files that are not currently in use, this occurs without requesting the user to re-boot.

10 For such files that are currently in use, the user is requested to re-boot, and the operating system installs the one or more bytes during the re-boot.

Fig. 6 illustrates another embodiment of a computer network generally designated 410. Network 410 comprises a selection server 412 with a data base 413, a content server 417, clients 414-416, and Internet 20 to interconnect the servers and

15 clients. By way of example, the selection server 412 comprises a computer workstation with associated programming and a modem, the content server 417 comprises a computer workstation with associated programming and a modem, and each of the clients 414-416 comprises a personal computer with associated programming and a modem.

20 Communications between the clients and the selection server 412 use hypertext transport protocol (HTTP) and communications between the clients and the content server 417 use file transport protocol (FTP). Although not shown, there are many other servers for the Internet.

Selection server 412 is dedicated to automating selection of data updates and additional data, collectively called "new data." The content server 417 includes the

25 actual, new data (stored on disk 421), and data base 413 includes meta data for the new data and FTP addressing information to locate the new data in content server 417. The meta data provides descriptive information about the respective new data such as those user attributes for which the new data is appropriate (assuming the user does not already have the new data). The FTP addressing information indicates the address or

30 identification of the content server which stores the respective new data (content server 417 in the illustrated example), a user ID and password to log-on to the content server, a

name of the new data, a list of files which comprise the new data and the size, checksum, and directory of each file.

According to one process for storing the new data in the content server 417 and the meta data in the selection server 412, a person writes the new data and meta data and loads them into the content server 417, either via the Internet or by manual loading. The new data remains stored in the content server, but the meta data is written to the selection server 412, either via the Internet or by manual loading.

As illustrated in Fig. 7, the selection server 412 is an HTTP server with an associated "CGI" program manager 431 and includes a data selection program 430. The client 414 includes an update manager program 432 (including a GUI), a scout routine 433, a service application routine 434, and a download routine 439. The functions of the server and client programming and a typical client-server session for selecting and downloading new data are illustrated in Figs. 8a and 8b.

A user at client computer 414 selects an icon to invoke update manager 432. In response, the update manager contacts the general manager 431 in server 412 to begin a session and supplies the current level of update manager 432, scout 433, service application 434, and download routine 439 within client 414 (step 504). In response, the general manager 431 determines if the client's update manager 432, scout 433, service application 434, and download routine 439 are the latest version (step 506). This determination is performed by comparing the client level information supplied by the client 414 to the data in selection server 412 for the latest version of the client's update manager 432, scout 433, service application 434, and download routine 439 stored in the content server 417.

If the client's update manager, scout, service application, and download routine are not the latest version, the server sends the FTP addressing information to the client to allow the client to retrieve the latest versions from a content server. Along with the FTP addressing for the client update manager, scout, service application, and download routine, the server sends FTP addressing information to the client for a serial number recognizer program 441 stored in content server 417 which will query client 414 for the serial number of client 414 (step 507). Next, the client downloads and installs the

latest version of the client update manager, scout, service application, and download routine from the content server if the FTP addressing was provided (step 508).

The furnishing of the FTP addressing information for the recognizer program 441 begins the new data selection process. The client next downloads the 5 recognizer program 441 from the content server and executes it (step 509). The recognizer program 441 then obtains the serial number of the client 414 from the basic input output system (BIOS) firmware, using scout APIs (step 509). The update manager routine 432 then sends the serial number of the client 414, obtained by the recognizer program 441, to the selection server 412 (step 510).

10 This initiates the selection update program 430 (step 512). The selection update program 430 within server 412 then determines, from a table 435, the employee number or name of the user that owns the machine comprising client 414. Alternately, when there is more than one user of client 414, the users send their names or serial numbers to the server 412 (in step 510). In this scenario, selection server 412 does not 15 send the FTP of recognizer program 441 to client 414 in step 507 and client 414 does not fetch or execute recognizer program 441. There is no need.

Once selection server 412 knows the employee number or name of the user of client 414, either from recognizer program 441 and table 435 or directly from the user, the selection update program can determine which data is appropriate for the user 20 and which data is not appropriate for the user. This eliminates the vast majority of the data stored in content server 417 (and other content servers, if any) from further consideration (step 514). In the illustrated embodiment, this determination is made by having the selection update program first read from table 435 the user attributes corresponding to the user's serial number or name. These user attributes may include the 25 user's job title, department, profession, age, income, or interests previously expressed by the user and forwarded to the server. Then, the selection update program correlates the meta data in data base 413 to the user's attributes. Matches between the meta data and the user's serial number or other attributes indicate that the corresponding new data may be appropriate for client 414. In practice, the meta data and table 435 can be arranged in 30 a relational data base to facilitate the correlation function.

For example, the following table lists data available in content server 417 and the user attributes-meta data that matches each of the new data.

	<u>New Data</u>	<u>User Attributes-Meta Data</u>
	Customer List	Marketing person
5	Price List	Marketing person
	Salary Chart	Manager
	Company Rules	All Employees
	Golf Ad #1 (news)	Golfer
	Golf Ad #2 (equipment)	Golfer
10	Bowling Ad #1	Bowler
	Bowling Ad #2	Bowler

In this example, users who are marketing employees are entitled to receive only "Customer List," "Price List," and "Company Rules" data. Users who are management employees are entitled to receive only "Salary Chart" and "Company Rules" data. Users who are golfers, but not employees, are targeted to receive only "Golf Ad #1" and "Golf Ad #2" data. Users who are bowlers, but not employees, are targeted to receive only "Bowling Ad #1" and "Bowling Ad #2" data.

For each new data which is consistent with the user's attribute(s), the selection update program 430 sends to the client 414 the FTP addressing information of a corresponding recognizer program and the corresponding meta data (step 516). For example, for a user having the attribute of golfer, the selection update program 430 sends the FTP addressing for recognizer programs 440 and 442, respectively for "Golf Ad #1" and "Golf Ad #2." The FTP addressing information is not provided to client 414, however, for the recognizer programs for the other new data which is not consistent with the user's attribute(s). From the FTP addressing information, the client 414 downloads the actual recognizer programs from the content server 417 (step 518).

Each recognizer program is executable and small, typically 2,000 - 3,000 bytes. Consequently, each recognizer program can be downloaded from content server 417 to the client 414 in seconds (whereas the corresponding new data may be much larger and take minutes or even hours to download).

It should be noted that, based on the user attribute(s) alone, the selection update program 430 within server 412 is not sure which of the new data that is consistent with the user's attribute(s) is actually needed by the client. For example, the client may already have some or all of the new data.

5 Next, the client 414 executes recognizer programs 440 and 442 with assistance from the scout 433 (step 518). The scout 433 assists the recognizer programs by providing subroutines that can be called by the recognizer programs. For files required by the recognizer programs, these callable subroutines can locate the files, determine the version of the files, determine the length of the file, identify a directory
10 which lists the files, obtain time stamps for the files, obtain data from inside the files, and obtain data from a program registry. This further reduces the requisite length of each recognizer program and therefore shortens the download time. When executed, each recognizer program further investigates storage of the client computer 414 to determine if the user already has the new data available from the client computer. Optionally, the
15 recognizer program may investigate a user profile 437 stored in the client computer 414 to determine whether the new data is consistent with other user attributes stored in user profile 437.

Recognizer program 440 is illustrated in more detail in Fig. 9 and investigates the user's storage as follows. In this example, the sole user attribute (stored
20 in table 435 in the selection server 412) is "golfer." First, recognizer program 440 determines from the scout routine 433 in client 414 if client 414 already has a copy of the Golf Ad #1 available for the user (decision 600). If not, then recognizer program 440 determines that, so far, "Golf Ad #1" is deemed appropriate for the user (step 602). If client 414 stores a file called "Golf Ad #1," however, then recognizer program 440
25 optionally will check if this is the same file as the one currently stored on the server with the same name. Accordingly, recognizer program 440 compares the size of the "Golf Ad #1" file stored in client computer 414 to the size of the "Golf Ad #1" file stored in the content server 417 (step 604 and decision 606). (The size of Golf Ad #1 stored in content server 417 was included in the meta data that was sent along with the FTP address of the
30 recognizer program.) If the sizes are different, then recognizer program 440 concludes that they are different files and that this file is deemed appropriate for downloading from the content server to the client computer 414 (step 602). If the sizes are the same, then

recognizer program 440 concludes that they are the same file, and another copy of "Golf Ad #1" is not appropriate for downloading to the client computer (step 608). Recognizer program 442 performs the same analysis for "Golf Ad #2."

For new data that are lengthy, a respective recognizer program could also
5 determine if the client computer has sufficient memory for the new data by using the appropriate scout function. A recognizer program could also compare the meta data for new data to user profile 437 maintained in client computer 414 to further determine whether the new data is appropriate for the user. For example, the user profile 437 may qualify that the user's interests in golf are limited to golf news but not golf equipment. In
10 such a case, the recognizer program would determine that "Golf Ad #1" relating to "news" would be appropriate for the user but "Golf Ad #2" relating to "equipment" would not be appropriate for the user.

Assume that another recognizer program 442 illustrated in Fig. 10 corresponds to the "Customer List" data. Recognizer program 442, when executing in
15 client computer 414, will determine if the user already has a file called "Customer List" (decision 700). If not, then the recognizer program 442 determines that this data is appropriate for downloading to the client (step 706). If the client already has a file called "Customer List," however, then recognizer program 442 will compare (step 704) the date associated with the creation of the "Customer List" file stored in the server (and fetched
20 with recognizer program 442) to the date associated with the creation of the "Customer List" file stored in client 414 (and obtained in step 702). If the date stored in the server is later, then the "Customer List" data stored in the server is appropriate for the user (step 706). Otherwise, it is not appropriate for the user (step 708).

The results of the recognizer programs include yes/no answers to the
25 question of whether each respective new data is appropriate for the user, i.e., not currently available in client 414 for the user and, optionally, consistent with the user's profile, if any, stored in table 435 in the client computer. (The selection update program previously determined that the new data were consistent with the user's attributes stored in table 435 in the server computer.) The recognizer programs can also assign a "critical"
30 level to each new data based on the need of the user for the new data. For example, if the data are especially important such as the "Customer List" or the "Salary Chart," and the

user has no version or a very old version, then this data would be assigned a high level of criticality. After the recognizer programs are executed in client 414, client 414 sends to selection server 412 a list of the new data which is appropriate for the user (step 520).

Based on the information gathered by the recognizer programs, the
5 selection update program determines the level of criticality of the respective data and builds a selection form for display at the client (step 522). The selection form comprises a list of the new data that is consistent with the user's attributes and appropriate for the client 414, and a display of the following three selection choices (step 524):

1. critical data (to minimize the download time);
10
2. all new data that are appropriate for the user; and
3. client selection of appropriate new data - user selects particular new data from the list of all new data which are appropriate for the user.

Next, the user makes a selection, either selection of the first or second
15 categories or an itemized list of new data pursuant to the third category (step 530), and sends the selection to the server (step 531). In response, the server 412 sends to the client 414 the FTP addressing information for the selected new data (step 532). Using this FTP addressing information, the download routine 439 of the client downloads the new data from the content server 417 (step 533). In a preferred embodiment, the FTP addressing
20 information for all new data represented in the selection form are passed with the selection form to the client 414 (in step 524).

As the new data are being downloaded, the download routine also records status information (such as the FTP addresses) of the new data in case of interruption to the communication line. If there is such an interruption, the downloading can continue
25 later where it left off. Upon completion of the download, the service application routine of client 414 installs the new data in the client 414 for access by the user as follows (step 534). For all files constituting the new data, except those which represent updates to existing data stored in the client computer for which the stale version is currently in use

by client 414, the service application routine replaces the existing files in client 414 with the new file. The update manager also lists each data file whose stale version is currently in use. Then, the user is requested to re-boot, and the operating system installs the listed data during the re-boot. For all appropriate new data selected by the user that represents
5 one or more bytes of a file, the service application routine replaces just the one or more bytes of the file. For such files that are not currently in use, this occurs without requesting the user to re-boot. For such files that are currently in use, the user is requested to re-boot, and the operating system installs the one or more bytes during the re-boot.

10 Techniques for selecting and transferring code updates and new data to a client computer have been disclosed. Numerous modifications and substitutions can be made, however, without deviating from the scope of the present invention. For example, there can be multiple content servers, or the selection server functions and content server function can be merged into one server. If desired, protocols other than HTTP and FTP
15 can be used. Further, a three-tiered architecture for determining whether to provide a software update may be implemented.

Figs. 11 and 12 are block diagrams showing another exemplary embodiment of a system and method in accordance with the present invention. The present invention determines whether to provide a software program update 820 (or a
20 new software program) to one of a plurality of client processors 860A-860N, each client processor 860A-860N requiring one of a plurality of software program updates. Fig. 11 shows an exemplary system in accordance with this embodiment of the invention.

The embodiment includes a three-tiered architecture for determining the need for, and controlling the application or removal (installation and uninstallation) of,
25 computer readable binary data (referred to as "updates") from a centrally administered location. The third (or highest) tier includes a selection server processor 850 (which may also be the content server providing mass storage for software and data). The second (or middle) tier includes an administration server processor 800A which may include a GUI. Administration server processor 800A communicates with selection server processor 850
30 on behalf of the client processors 860A-860N, the first (or bottom) tier processors. The administration server processor 800A controls which updates are installed and uninstalled

on the client processors 860A-860N. The client processors 860A-860N each include a "client agent," a program executed by each client processor which accepts commands from administration server processor 800A, processes the commands, and returns status information 870, 871 (described in detail below) to administration server processor 800A.

5 In the exemplary system, at least one selection server processor 850 provides a source of software program updates 820 for client software programs and system configuration management tools, which are described in detail below. The selection server processor 850 may be, for example, an enterprise server which serves an entire business or university, or a division or subset of a larger entity. Alternatively, the
10 selection server processor 850 may be a server processor for a software vendor; a plurality of administration server processors 860A-860N may communicate with the selection server processor 850 via a worldwide telecommunications network, such as the Internet.

15 At least one administration server processor 800A is provided.
Administration server processor 800A provides the tools and resources for managing the system configuration in the client processors 860A-860N served by administration server processor 800A. The administration server processor 800A and client processors 860A-860N may, for example, all be connected in a local area network (LAN) 860 within an office, a floor of a building, or a local campus. On the other hand, the administration
20 server may serve a plurality of client processors that are related by business function; the client processors 860A-860N and administrative server processor 800A need not all be co-located.

The administration server processor 800A and its GUI have the following general features:

25 1) The ability to logically group multiple first-tier client processors in user definable groups.
2) The ability to log (track) activity corresponding to each first-tier client processor.

3) The ability to perform the "update discovery" task for groups or single first-tier client processors.

4) The ability to perform the "update apply" task for sets of updates onto groups or single first-tier client processors.

5 5) The ability to perform the "update remove" task for sets of updates from groups or single first-tier client processors.

6) The ability to schedule the "update discovery," "update apply," and "update remove" tasks to occur at any given time and frequency.

10 7) The ability to view which updates are potentially applicable to each individual client processor or group of client processors.

8) The ability to view which client processors are in need of which updates or a group of updates.

15 Although FIG. 11 only shows a single LAN 860, each of the other administration server processors 800B and 800C has one or more client processors (not shown). The description of exemplary administration server processor 800A and client processors 860A-860N also applies to administration server processors 800B and 800C and their respective client processors (not shown).

20 FIG. 12 shows in greater detail the pertinent processes and data flows within the exemplary system. In particular, the functions and data flows of administration server processor 800A are emphasized in FIG. 12.

25 The administration server processor 800A includes three major tasks described below: the "update discovery," "update apply," and "update remove" tasks. Within the administration server processor 800A, the "update discovery" task includes blocks 810-816. The "update apply" task includes blocks 817 and 818. The "update remove" task is not shown in FIG. 12. The blocks 810-818 within administration server processor 800A include both general purpose computer hardware and software program

DETAILED DESIGN

instructions in administration server processor 800A, including conventional intra-processor and inter-processor communications functions.

Update Discovery

The “update discovery” task is defined as the administration server process of determining the applicability of (or need for) a set of updates 820 on a group of (or single) client processors 860A-860N. This process consists of communicating to selection server processor 850 in order to identify the administration server processor 800A and its level and type. This communication may be performed, for example, by using HTTP.

10 The selection server processor 850 returns meta data (which may be HTTP links) directing or pointing the administration server processor 800A to download a “system recognizer” program 851. The administration server processor 800A downloads (for example, via FTP) the system recognizer program 851 from the selection server processor 850.

15 The system recognizer program 851 is provided by selection server processor 850 to a system recognizer receiving means 810 in administration server processor 800A. System recognizer transmitting means 811 transmits the system recognizer program 851 to the client processor 860A. At block 862, the client processor 860A executes the system recognizer program 851 to determine the system configuration attributes.

20 The administration server processor 800A communicates a command and meta data (via HTTP) to each client processor 860A-860N, as directed by the system administrator (user) from the GUI of the administration server processor 800A. This command and meta data cause each affected client processor 860A-860N to download (via HTTP) the system recognizer program 851 and execute it. The system recognizer program 851 is responsible for determining a set of hardware and/or software attributes for each client processor 860A-860N that executes the system recognizer program 851. The system recognizer program 851 in each client processor 860A-860N then returns the

results (in the form of a set of system configuration attributes 870) to the administration server processor 800A.

Administration server processor 800A includes configuration receiving means 812 for receiving and storing a respective set of system configuration attributes 870 from each one of the plurality of client processors 860A-860N. The system configuration attributes 870 indicate whether a copy of any of the client software programs may be stored in the respective client processor 860A-860N. Administration server processor 800A also includes configuration transmitting means 813 for transmitting each set of system configuration attributes 870 to selection server processor 850.

This high-level identification of the configuration of each client processor may be performed at the time the system is set up, or infrequently thereafter. The high-level identification using the system recognizer program 851 need not be repeated every time the "update discovery" task is performed. Optionally, the system recognizer program 851 may be repeated every time the "update discovery" task is performed.

The selection server processor 850 then returns, to the update receiving means 814 of administration server processor 800A, meta data describing all the potential updates stored in selection server processor 850 that may apply to any client processor 860A-860N with the attributes 870 which have been transmitted to selection server 850.

The meta data include information directing (pointing) to an "update recognizer" program 852 for each identified potential update 820. The update receiving means 814 of administration server processor 800A then downloads software 853, including the updates 820 and update recognizer program 852 (via FTP), from the selection server processor 850 to the administration server processor 800A. The update receiving means 814 receives a respectively different update recognizer program 852 and software program update 820 from the selection server processor 850 corresponding to each respective one of the plurality of client software programs.

An update recognizer transmitting means 815 transmits to each client processor 860A-160N the update recognizer program 852 corresponding to the respective client software programs in each respective client processor 860A-160N. Each client

processor (e.g., 860A) executes at least one of the update recognizer programs in block 864 to issue a notification 871 to administration server processor 800A. The notification 871 indicates whether the corresponding software program update 820 is applicable to the copy of the respective client software program in that client processor 860A. In the 5 exemplary embodiment, the administration server processor 800A communicates a command and meta data (via HTTP) to each client processor 860A with the corresponding system attributes. The command and meta data cause each affected client processor 860A to download (via HTTP) the update recognizer 852, execute it, and return the results to the administration server processor 800A.

10 As described above, the system recognizer program 851 provides high level information about the hardware and/or software configuration of a client processor 860A sufficient for a coarse level determination of which updates may potentially be applicable to a given client processor 860A. The update recognizer 852 is responsible for determining the actual applicability (need) of the specific update to the specific client 15 processor 860A with a fine degree of specificity.

For example, the system recognizer 851 may identify which programs reside in each client processor. The update recognizer program 852 would identify the specific level (version, release, and/or maintenance level) of the copy of the program in the client processor 860A. The update recognizer 852 would determine whether a 20 specific update should be installed based on the actual level of the copy of the program in the client processor 860A.

The update recognizer program 852 is a relatively short agent program which can make very specific decisions about which updates should be applied to an individual client processor system 860A. This provides a fine level of granularity in 25 configuration management, by allowing the updates themselves to decide whether they apply to the system. The small update recognizer program that comprises the agent for the update may be written by the update author, or by another similarly qualified expert knowledgeable of the criteria for determining which update should be installed in which systems.

DRAFT - 06/2000

Administration server processor 800A includes notification receiving means 816 for receiving and storing the respective notification 871 from each client processor 860A-860N. After the update recognizers are run on all affected client processors 860A-860N, the resulting information may be stored by the administration 5 server processor 800A. The administration server processor 800A then displays via its GUI which updates are applicable to which client processor 860A or group of clients.

Update Apply

The “update apply” task is defined as the administration server process of installing a set of updates 820 onto a client processor 860A or group of clients 860A-10 860N. A selecting means 817 is provided for selecting one of the plurality of software programs and one of the plurality of clients. The administration server GUI provides a selecting means 817 to select (choose) which updates are to be applied to which client 860A or group of clients 860A-860N. When the “update apply” process is invoked for a set of client processors 860A-860N, only those updates which are selected (chosen) by 15 the operator are actually installed.

The selecting means 817 may use an additional, user-defined criterion other than a hardware configuration or software configuration of the plurality of client processors 860A-860N to perform the selecting. For example, the selection server processor 850 may have one hundred updates. Based on the configuration attributes for 20 the client processors on a given LAN, only ten of the one hundred updates may be potentially applicable for the client processors of the LAN. Further, execution of the update recognizer programs associated with these ten updates may result in identification of only six updates that should specifically be installed on the LAN. (For example, the client processor may already have a higher release level for the remaining four.) Of these 25 six applicable updates, the system administrator (operator) may choose to install only two updates on a given occasion, for business reasons unrelated to the system configuration of the client processors. For example, updates may be delayed for budgetary reasons, or for the convenience of a specific business group or department of client processor users.

One of ordinary skill in the art recognizes that business-related criteria for 30 deciding when to update software (which are independent of the hardware and software

configuration criteria) may be included in the selecting means 817 of the administration server processor software. For example, the selecting means 817 may include program code to: (1) categorize all the client processors 860A-860N by business group or function; (2) coordinate the schedules of the users of the various client processors 860A-860N; and (3) select a time for installing updates 820, so as to minimize disruption of user activity.

Once the selection is made, an update transmitting means 818 is responsive to the selecting means 817 for transmitting a software program update 820 associated with the selected software program to the selected client processor 860A. The administration server processor 800A downloads (for example, via FTP) all of the necessary components (files) from the content server (i.e., selection server 850) for each update 820 that is to be applied. The components that are to be downloaded to administration server processor 800A are extracted from the meta data received from the selection server 850 during the "update discovery" process, described above.

After downloading the updates 820 to the administration server processor 800A, the update transmitting means 818 transmits the update 820 to the selected client processors 860A-860N in which the updates 820 are to be installed. To accomplish this, the administration server processor 800A sends a command and meta data to each selected client processor 860A-860N for each update 820 that is to be applied. The command causes the recipient client processor 860A to download (e.g., via HTTP) the necessary components (files) and execute the respective "update installer" for each component, to appropriately install the update 820. Upon completion, a status message (not shown) is sent from the client processor 860A to the administration server processor 800A.

After the status message is received by administration server processor 800A, indicating installation on a client processor 860A, administration server processor 800A updates its configuration data to indicate that the update 820 is no longer available for installation in that client processor 860A, and that the update is now available for removal (uninstall). In some cases, of course, the update may not be removable.

Update Remove

The “update remove” task (not shown) is defined as the administration server processor 800A process of uninstalling a set of updates 820 from a client processor 860A or group of clients in a manner analogous to the “update apply” task described above. In the case of the “update remove” task, the command and corresponding action 5 uninstalls the update. After removal from a client processor 860A, the update 820 is no longer available for removal, but is again available for installation.

Other Embodiments and Variations

In the exemplary embodiment, the selection server processor 850 is also 10 described as being the content server for mass storage of software and data. One of ordinary skill in the art recognizes that a selection server processor according to the invention may also be instantiated in a separate and distinct processor from the content server processor. In this variation, the selection server would transmit meta data 15 (including pointers) to the administration server processor 800A. The meta data would direct the administration server processor 800A to the location of files in the content server processor.

In the exemplary embodiment, the administration server processor 800A employs a GUI for ease of use. One of ordinary skill in the art recognizes that an 20 administration server processor according to the invention may also use a text base user interface.

Further, the invention may also be embodied in the form of program code embodied in tangible media, such as random-access memory (RAM), read-only memory (ROM), floppy diskettes, CD-ROMs, hard disk drives, zip drives, or any other machine-readable storage medium, wherein, when the program code is loaded into and executed 25 by a machine, such as a computer, the computer becomes an apparatus for practicing the invention. The invention may also be embodied in the form of program code, for example, whether stored in a storage medium, loaded into and/or executed by a machine, or transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via electromagnetic radiation, wherein, when the program code is

loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the invention. When implemented on a general-purpose processor, the program code segments combine with the processor to provide a unique device that operates analogously to specific logic circuits.

5 Although illustrated and described herein with reference to certain exemplary embodiments, the present invention is nevertheless not intended to be limited to the details shown. Rather, various modifications may be made in the details within the scope and range of equivalents of the claims and without departing from the spirit of the invention.